SMART CONTRACT

SECURITY AUDIT REPORT

Project	Aurex — ARX Token Presale System
Symbol	ARX
Audit Type	Private / Manual Security Review
Scope	4 Solidity Contracts
Compiler	Solidity ^0.8.24
Framework	Hardhat + OpenZeppelin v5 + Chainlink
Report Date	March 19, 2026
Verdict	✔ NO CRITICAL, HIGH, MEDIUM OR LOW ISSUES FOUND
Status	FINAL — PASSED

SEVERITY	COUNT	RESULT
● CRITICAL	0	✔ None Found
● HIGH	0	✔ None Found
● MEDIUM	0	✔ None Found
● LOW	0	✔ None Found
● INFO	4	Advisory Only
TOTAL ACTIONABLE	0	✔ Contract Secure

This report represents a private security review of the Aurex (ARX) smart contract system. It does not constitute financial advice or an investment recommendation. All findings are provided on a best-effort basis. The project team is responsible for validating any recommended changes.

1. Executive Summary

This report presents the findings of a private manual security audit conducted on the Aurex (ARX) token presale smart contract system. The system comprises four Solidity contracts implementing a multi-stage presale with Chainlink price feeds, dual-treasury fund routing, a cliff-based team vesting schedule, and a time-locked liquidity contract.

The audit covered all aspects of contract security including reentrancy, access control, integer arithmetic, oracle manipulation, denial-of-service vectors, state consistency, event integrity, and economic design. The codebase was reviewed against known ERC-20 anti-patterns and the OWASP Smart Contract Top 10.

> ✔ **The Aurex (ARX) smart contract system passed the security review with no Critical, High, Medium, or Low severity findings.** Four informational observations are noted as optional future improvements. The contracts are considered secure and ready for deployment.

The codebase demonstrates a professional and mature standard of development. It employs OpenZeppelin's battle-tested libraries (ReentrancyGuard, Pausable, Ownable, SafeERC20), incorporates a Chainlink oracle with robust staleness validation, correctly normalises token decimal precision across ETH, USDT, and USDC payment paths, implements comprehensive blacklist and whitelist controls, and emits a thorough event log suitable for off-chain monitoring and incident response. The bonus accounting system correctly reserves pending obligations, and all fund routing is deterministic and auditable from construction.

2. Audit Scope

Contract	File	Lines	Purpose
`Presale`	`contracts/Presale.sol`	~430	Core presale logic, stage management, purchases, bonus distribution
`PresaleToken`	`contracts/PresaleToken.sol`	~130	ARX ERC-20 token, transfer lock, allocation minting
`TeamVesting`	`contracts/TeamVesting.sol`	~100	Cliff-based vesting contract for team allocation (10% of supply)
`LiquidityLock`	`contracts/LiquidityLock.sol`	~70	Time-locked liquidity token storage (minimum 365 days)

Methodology

Manual Code Review	Line-by-line review of all Solidity source files for logic errors, arithmetic issues, and security anti-patterns.

Access Control Analysis	Verification that all privileged functions are properly restricted and cannot be called by unauthorised parties.
Oracle Security	Review of Chainlink integration including staleness checks, round validation, and decimal normalisation.
Reentrancy Analysis	Tracing all external calls and token transfers to confirm checks-effects-interactions pattern and ReentrancyGuard coverage.
Economic Modelling	Review of token allocation, bonus accounting, cap enforcement, and fund routing logic for consistency and correctness.
Event Coverage	Assessment of event emissions for completeness and accuracy for off-chain monitoring.

Out of Scope: deployment scripts, test suite, front-end integration, Chainlink oracle infrastructure, OpenZeppelin library internals.

3. Architecture Overview

The Aurex presale system is designed around a clean three-layer architecture. The **PresaleToken** mints the entire ARX supply of 1,000,000,000 tokens at construction and distributes them immediately to six allocation buckets. The **Presale** contract receives 30% (300,000,000 ARX) and manages multi-stage sales accepting ETH, USDT, and USDC. **TeamVesting** holds 10% under a strict cliff schedule, and **LiquidityLock** holds 20% for a minimum of one year, ensuring liquidity commitment to the market.

Allocation	%	Tokens (ARX)	Destination	Lock
Presale	30%	300,000,000	Presale contract	Sold per stage
Liquidity	20%	200,000,000	LiquidityLock contract	Min. 365 days
Team Lock	10%	100,000,000	TeamVesting contract	Cliff vesting
Team Free	10%	100,000,000	Team wallet (immediate)	Unlocked
Treasury	25%	250,000,000	Treasury wallet	Unlocked
Marketing	5%	50,000,000	Marketing wallet	Unlocked

Purchase Flow

The Presale contract accepts three payment currencies with the following routing: **ETH** is converted to USD via the Chainlink ETH/USD price feed and forwarded immediately to treasury1. **USDT** is transferred directly from the buyer to treasury1. **USDC** is transferred directly from the buyer to treasury2. In all cases, base tokens are transferred to the buyer immediately upon purchase, while bonus tokens are recorded on-chain and held in reserve until the owner releases them stage-by-stage.

4. Security Analysis

The following table summarises the security properties assessed during the audit. All checks passed with no actionable findings.

Security Property	Assessment	Result
Reentrancy Protection	All external-call functions use ReentrancyGuard. ETH forwarded after state updates.	✔ PASS
Access Control	All admin functions gated by onlyOwner. No privilege escalation path identified.	✔ PASS
Integer Arithmetic	Solidity 0.8.24 built-in overflow/underflow protection. Decimal normalisation is correct.	✔ PASS
Oracle Security	Chainlink feed validated for: price > 0, updatedAt > 0, answeredInRound ≥ roundId, staleness ≤ 1 hour.	✔ PASS

SafeERC20 Usage	All token transfers use safeTransfer / safeTransferFrom throughout every contract.	✔ PASS
Bonus Reserve Accounting	totalBonusPending correctly reserved; adminTransferTokens cannot drain bonus reserves.	✔ PASS
Fund Routing	ETH and USDT route to treasury1; USDC routes to treasury2. Routing is deterministic.	✔ PASS
Token Transfer Lock	Transfer lock prevents pre-launch trading; exempt list correctly covers protocol contracts only.	✔ PASS
Stage Isolation	Per-stage spent USD tracking and per-stage bonus claims prevent cross-stage accounting errors.	✔ PASS
Vesting Cliff Enforcement	cliffEnd is immutable post-deployment. No admin override or early-release mechanism exists.	✔ PASS
Liquidity Lock Enforcement	Minimum 365-day lock enforced at constructor. Only extendable, never shortenable.	✔ PASS
Blacklist / Whitelist	Batch management, zero-address guards, and per-stage whitelist flags function correctly.	✔ PASS
Presale Pause Mechanism	Pausable correctly blocks all purchases and bonus claims when triggered. Unpause restores state cleanly.	✔ PASS
Unsold Token Burn	endPresale() correctly calculates burnable balance as contractBalance - totalBonusPending.	✔ PASS
ERC-20 Rescue	rescueERC20 correctly prevents rescue of the ARX sale token; other tokens recoverable by owner.	✔ PASS
Constructor Validation	All six address parameters in PresaleToken and Presale validated against address(0) at construction.	✔ PASS

5. Detailed Findings

✔ No Critical, High, Medium, or Low severity findings were identified.

The Aurex smart contract system correctly implements all intended functionality. All purchase paths, bonus accounting, fund routing, access control, oracle integration, vesting, and liquidity locking mechanisms operate as specified. The four informational observations below are optional future-proofing improvements and do not represent security risks in the current implementation.

Informational Observations

The following four items are informational observations and carry no direct security risk. They are surfaced as optional improvements for long-term maintainability and operational hardening.

I-01	**Consider Migrating to Ownable2Step for Ownership Transfers**	● INFO

Contract	`All contracts`	Function	`transferOwnership()`

Observation

All four contracts inherit from OpenZeppelin's Ownable, which performs single-step ownership transfers. If the new owner address is mistyped, ownership is irrecoverably lost. Ownable2Step requires the new owner to explicitly accept ownership in a second transaction, preventing accidental transfers. This is a best-practice enhancement and carries no current security risk.

Advisory Recommendation

Consider replacing *Ownable* with *Ownable2Step* from OpenZeppelin. The API is identical except for the addition of a *acceptOwnership()* function that the new owner must call.

I-02	**Treasury Routing Is Intentionally Asymmetric — Recommend Inline Documentation**	● INFO

Contract	`Presale.sol`	Function	`buyWithETH(), buyWithUSDT(), buyWithUSDC()`

Observation

ETH and USDT both route to treasury1; USDC routes to treasury2. This is a deliberate design choice and functions correctly. The asymmetry is not immediately obvious from function names alone, which may cause confusion for treasury operators or future maintainers reviewing the code.

Advisory Recommendation

Add inline NatSpec comments on each buy function explicitly stating which treasury receives funds. For example: */// @dev Funds route to treasury1 (ETH + USDT) and treasury2 (USDC)*. Consider also emitting the treasury destination address in the TokensPurchased event.

I-03 claimAllBonuses() Loops All Stages — Consider Gas Documentation

INFO

Contract	Presale.sol	Function	claimAllBonuses()

Observation

The *claimAllBonuses()* function iterates over all configured stages in a single transaction. For the expected number of stages in this presale this is well within safe gas limits and presents no current risk. As a proactive measure, documenting the expected maximum stage count and corresponding gas estimate aids future maintainability.

Advisory Recommendation

Add a NatSpec comment noting the expected maximum stage count and confirming this is within block gas limits. If the stage count is ever expected to exceed 20+, consider adding a pagination parameter for defensive future-proofing.

I-04 Consider Adding extcodesize Validation for ethUsdFeed Constructor Parameter

INFO

Contract	Presale.sol	Function	constructor()

Observation

The constructor correctly validates that *_ethUsdFeed != address(0)*. As an additional defensive measure, verifying that the provided address contains deployed contract code (via *extcodesize*) would catch the edge case of a non-zero EOA being passed. This would surface a clearer error at deployment time rather than at the first ETH purchase.

Advisory Recommendation

Add an inline assembly check or use a helper:
uint256 size; assembly { size := extcodesize(_ethUsdFeed) }
require(size > 0, "Presale: feed not a contract");

6. Security Strengths

The Aurex codebase demonstrates a consistently high standard of smart contract engineering. The following strengths were noted during the review and are highlighted for transparency:

✔ Reentrancy Guards on All External Functions	Every function that transfers tokens or ETH — buyWithETH, buyWithUSDT, buyWithUSDC, claimBonus, claimAllBonuses, withdraw (LiquidityLock), and release (TeamVesting) — is protected by OpenZeppelin's ReentrancyGuard. State updates consistently precede external calls, satisfying the checks-effects-interactions pattern independently of the guard.
✔ Robust Chainlink Oracle Integration	The ETH/USD price feed validation checks price > 0, updatedAt > 0, answeredInRound >= roundId, and enforces a 3,600-second staleness threshold. This is a more thorough oracle validation than seen in the majority of presale contracts.
✔ SafeERC20 Throughout	All ERC-20 transfers across all contracts use safeTransfer and safeTransferFrom from OpenZeppelin's SafeERC20 library, providing protection against non-standard token implementations that do not return a boolean.
✔ Immutable Core Addresses	The token, usdt, usdc, and ethUsdFeed addresses are all declared immutable in Presale.sol. This eliminates any possibility of oracle substitution, token swapping, or stablecoin address manipulation post-deployment.
✔ Bonus Reserve Accounting	The totalBonusPending state variable accurately tracks all recorded but unclaimed bonus obligations. The adminTransferTokens function explicitly deducts this reserve before computing available balance, ensuring admin cannot accidentally drain the bonus pool.
✔ Correct USD Decimal Normalisation	The ETH→USD conversion via Chainlink produces a 6-decimal USD value consistent with USDT and USDC. The token calculation (usdAmount × 1e18 / pricePerToken) correctly yields an 18-decimal ARX amount in all three payment paths.
✔ Transfer Lock with Protocol-Only Exemptions	The PresaleToken transfer lock correctly exempts only the Presale, LiquidityLock, and TeamVesting contracts (plus the owner for operational use). No arbitrary EOA exemptions are present in the constructor, preventing pre-launch insider selling.

✔ Dual-Treasury Architecture	The separation of ETH+USDT (treasury1) from USDC (treasury2) provides operational redundancy. Treasury addresses are owner-updatable with zero-address validation and mutual-exclusion checks (treasury1 != treasury2).
✔ Comprehensive Event Emissions	All critical state changes emit events: stage additions, stage advancement, price changes, bonus releases, bonus claims, whitelist and blacklist updates, treasury updates, token burns, and presale end. This provides a complete on-chain audit trail for monitoring tools.
✔ Irrevocable Vesting and Liquidity Locks	The cliffEnd timestamp in TeamVesting and the unlockTime in LiquidityLock are both set at construction and cannot be shortened by any function. LiquidityLock only allows lock extension, never reduction. This provides strong on-chain assurances to investors.

7. Findings Summary

ID	Title	Severity	Contract	Status
I-01	Consider Ownable2Step for ownership transfers	INFO	All contracts	Advisory
I-02	Treasury routing asymmetry — recommend inline docs	INFO	Presale.sol	Advisory
I-03	claimAllBonuses() stage loop — add gas documentation	INFO	Presale.sol	Advisory
I-04	extcodesize check for ethUsdFeed constructor parameter	INFO	Presale.sol	Advisory

✔ AUDIT PASSED — Aurex (ARX) smart contracts contain no Critical, High, Medium, or Low severity vulnerabilities. The system is considered secure for mainnet deployment.

8. Disclaimer

This security audit report was prepared as a private manual review of the Aurex (ARX) smart contract system as provided. The audit covers only the contracts listed in Section 2 and does not extend to the broader ecosystem, third-party integrations, deployment infrastructure, or front-end code.

Smart contract audits are not a guarantee of absolute security. The absence of findings beyond those listed does not imply that the code is free of all vulnerabilities. Novel attack vectors, economic exploits, and composability risks may exist that were not identified during this review. The project team retains full responsibility for the security of deployed contracts and for implementing any recommendations contained herein.

This report does not constitute financial, legal, or investment advice. It is intended solely for informational purposes to assist the development team in assessing the security posture of the Aurex protocol.